面向对象设计的4个特点：抽象，封装，继承，多态

程序=算法+数据结构

**类和对象（student 类）**

1.构造函数

2.析构函数

3.拷贝构造函数

4.运算符重载

5.对象指针

@指向对象成员的指针

student s;

student \*p=&s;

@指向对象数据成员的指针

int \*q=&s.num；

@指向对象成员函数的指针

void（student::\*t)()=&student::print;

@this指针

6.对象数组

student s[10];

7.共用数据的保护

@常对象 ：类名 const 或 const 类名

@常对象成员（常数据成员只能用初始化列表初始化和常成员函数）

@指向对象的常指针：类名 \* const 指针变量名

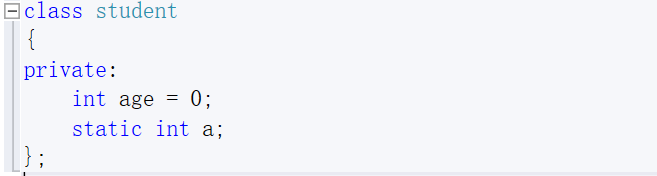
@指向常对象的指针变量 const 类型名 \* ptr

8.静态数据成员（关键字 static）

①静态数据成员独占一份空间，所有对象共享

②静态数据成员必须在类外初始化

③C++11支持非静态成员变量在声明时，直接初始化。



9.静态成员函数（没有this指针)

10.内置成员函数：为了减少时间开销，如果在类体中定义的成员函数不包括循环等控制结构，C++系统自动地对他们作为内置函数（inline）来处理。

11、友元

友元分为：友元函数和友元类

友元提供了一种突破封装的方式，有时提供了便利。但是友元会增加耦合度，破坏了封装，所以友元不宜多用。

11.1 友元函数：友元函数可以直接访问类的私有成员，它是定义在类外部的普通函数，不属于任何类，但需要在类的内部声明，声明时需要加friend关键字。

说明:

1. 友元函数可访问类的私有成员，但不是类的成员函数
2. 友元函数不能用const修饰
3. 友元函数可以在类定义的任何地方声明，不受类访问限定符限制
4. 一个函数可以是多个类的友元函数
5. 友元函数的调用与普通函数的调用和原理相同

11.2 友元成员函数：friend函数不仅可以是一般函数（非成员函数），而且可以是另一个类中的成员函数。注意：可能会用到**类的提前引用声明。**

#include <iostream>

using namespace std;

class Date; //对Date类的提前引用声明

class Time

{

public:

Time(int hour, int minute, int second)

:hour(hour)

, minute(minute)

, second(second)

{}

void display(Date& d); //引用Date类的对象

private:

int hour;

int minute;

int second;

};

class Date //Date类的声明

{

public:

Date(int year, int month, int day)

:year(year)

, month(month)

, day(day)

{}

void Time::display(Date& d) //定义display函数，只能在Date类后面定义

{

cout << d.year << "-" << d.month << "-" << d.day << " " << hour << ":" << minute << ":" << second << endl;

}

int main()

{

Time t(12, 30, 8);

Date d(2019, 12, 23);

t.display(d);

return 0;

}

11.3 友元类

①友元类的所有成员函数都可以是另一个类的友元函数，都可以访问另一个类中的非公有成员。

②友元关系是单向的，不具有交换性。

比如上述Time类和Date类，在Time类中声明Date类为其友元类，那么可以在Date类中直接访问Time类的私有成员变量，但想在Time类中访问Date类中私有的成员变量则不行。

③友元关系不能传递

如果B是A的友元，C是B的友元，则不能说明C时A的友元。

友元利弊：友元是对面向对象的封装性进行破环，但是有助于数据共享，能提高程序的效率。使用时做好权衡！

12、内部类

概念：如果一个类定义在另一个类的内部，这个内部类就叫做内部类。注意此时这个内部类是一个独立的类，它不属于外部类，更不能通过外部类的对象去调用内部类。外部类对内部类没有任何优越的访问权限。

注意：内部类就是外部类的友元类。注意友元类的定义，内部类可以通过外部类的对象参数来访问外部类中的所有成员。但是外部类不是内部类的友元。

特性：

①内部类可以定义在外部类的public、protected、private都是可以的。

②注意内部类可以直接访问外部类中的static、枚举成员，不需要外部类的对象/类名。

③sizeof(外部类)=外部类，和内部类没有任何关系。